MULTIMEDIA UNIVERSITY

STUDENT ID NO

TABLE NO: _____

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 3, 2015/2016

TCS1011 – DATA STRUCTURES AND ALGORITHMS

(All sections / Groups)

31 MAY 2016
9:00 am – 11:00 am
(2 Hours)

Question	Mark
1	
2	
3	
4	
Total	

INSTRUCTIONS TO STUDENT

1. This question paper consists of 8 printed pages (including the cover page) with four questions. Answer all questions.
2. Attempt all questions. Each question carries 10 marks.
3. Print all your answers **CLEARLY** in the specific answer box provided for each question. Submit this question paper at the end of the examination.

QUESTION 1 [10 marks]

(a) Below is a class declaration for a pointer based list. Implement the `display()` function which displays every node in the list.

```
class LinkedList
{
  public:
    LinkedList( );        // getLength() and size are not available
    ...                   // in this class.
    void display( );      // The last node points to NULL
    ...

  private:
    struct ListNode
    {
      int num;            // There are two pieces of data in each node
      char item;
      ListNode *next;
    }
    ListNode *head;
    ...
}
```

[4]

```
void LinkedList::display()
{




}
```

(b) Show the output of the following program.

```
#include <iostream>
#include "StackP.h"
using namespace std;
int main()
{
    Stack s;
    for ( int i = 10; i > 0; --i )
        s.push(i);

    while ( !s.empty() )
    {
        s.pop();
        if ( !s.empty() ) s.pop();
        if ( !s.empty() ) cout << s.top() << " ";
    }
    return 0;
}
```

[2]

Continued ...

(c) A pointer based Stack class declaration is shown in the following code. Write the code to complete the `push()` function. Assume that exception handling works and no additional code needs to be added for it.

[4]

```
...

typedef char StackItemType;

/** @class StackException
 * Exception class throw by ADT stack. */
class StackException : public logic_error
{
  public:
    StackException(const string& message = "")
            : logic_error(message.c_str())
    {}
};

class Stack
{
  public:
    Stack();
    ...
    bool isEmpty() const;
    void push(const StackItemType& newItem) throw(StackException);
    ...

  private:
    struct StackNode
    {
        StackItemType item;
        StackNode    *next;
    };
    StackNode *topPtr;
};

Stack::Stack() : topPtr(NULL) { }
...

bool Stack::isEmpty() const
{
    return topPtr == NULL;
}

void Stack::push(const StackItemType& newItem) throw(StackException)
{
    try
    {









    }
```

Continued ...

```
    catch (bad_alloc e)
    {
        throw StackException(
            "StackException: push cannot allocate memory.");
    }
}
```

QUESTION 2 [10 marks]

(a) Write the code of a recursive function which calculates the factorial of n.

[2]

```
int factorial(int n)
{



}
```

(b) Explain in your own words or pseudo-code how a postfix expression can be evaluated using a stack.

[4]

(c) A pointer based Queue class declaration is shown in the following code. Write the code to complete the `dequeue()` function.

[4]

Continued ...

```
...

typedef char QueueItemType;

class Queue
{
    public:
        Queue();
        bool isEmpty();
        void enqueue(QueueItemType newItem);
        void dequeue();

        ...

    private:
        struct QueueNode
        {
            QueueItemType item;
            QueueNode* next;
        };  // end QueueNode

        QueueNode* backPtr;
        QueueNode* frontPtr;

};

void Queue::dequeue()
{

    if (isEmpty())
    {
        cout << "dequeue: Queue is empty." << endl;
        return;
    }
    else
    {
        QueueNode* tempPtr = frontPtr;

        // Write your code from here









        tempPtr->next = NULL;
        delete tempPtr;

    }
}
```

QUESTION 3 [10 marks]

(a) Describe the running time of the following pseudo code in Big O notation.

Continued ...

```
for (int i = 0; i < n; ++i)
{
  for (int j = 0; j < n; ++j)
    a= b + c;
  for (int k = 0; k < n; ++k)
    cout << a + b;
}
```

[2]

(b) Implement the bubble sort algorithm by completing the `bubbleSort()` function. The `bubbleSort()` function sorts the items in an array into ascending order. The `swap()` function has been provided. There is no need to implement a `main()` function.

[5]

```
void swap(DataType& x, DataType& y)
{
   DataType temp = x;
   x = y;
   y = temp;
   cout << "Swapped " << x << " with " << y << " --->";
}

   //  Sorts the items in an array into ascending order.
   //  theArray is an array of n items.
   //  n  The size of theArray.

void bubbleSort(DataType theArray[], int n)
{









}
```

(c) The following is a sequence of numbers stored using the hash function `storeIndex = input % 12`, where `input` is the number to be hashed, and linear probing method is used to resolve collisions.

14, 5, 10, 2, 0, 200, 29, 15, 6, 8, 11, 31

Continued ...

Show the resulting storage map using the table provided below.

[3]

StoreIndex	Number
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

QUESTION 4 [10 marks]

(a) Complete the following pseudo code for a depth-first search traversal of a graph.

[3]

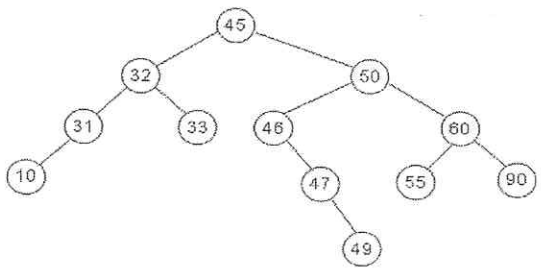
```
// Traverses a graph beginning at vertex v by using a
// recursive depth-first search
dfs( in v:Vertex )
{




}
```

(b) Draw the resulting binary search tree after nodes 31 and 50 are deleted from the following binary search tree.

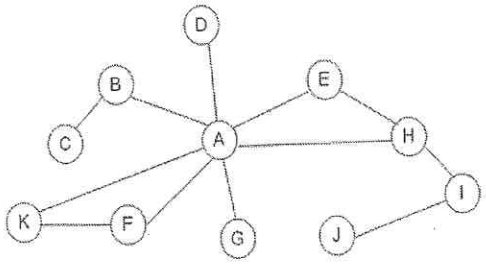
[2]



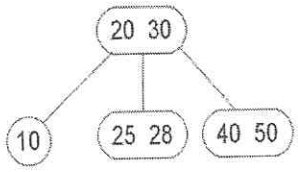
Continued ...

(c) Show the sequence of vertices visited if a depth-first search traversal is conducted starting from vertex A for the following graph. Choose in alphabetical order when there are more than one valid choices during the traversal.

[2]



(d) Draw the resulting 2-3 trees after nodes 8 and 26 are inserted one after another into the following 2-3 tree.



[3]

After 8 is inserted:

After 8 and 26 are inserted:

End of Page